

Work Directory (WD), History, Source, Comm

WD - is the computer folder with the file you read in or save out.

`getwd()` See current WD

`setwd()` Set, change WD

`dir()` Lists content of the dir

History

`savehistory(file="Name.Rhistory")`

`loadhistory(file="Name.Rhistory")`

`sink("name.txt")` `sink()`

open-close connection

Source

`source("CommandScript.R")`

Commands stored in scripts in WD

Created objects saved and loaded in the WD:

`save(object, file="path")`

`load(file="path")`

Commands

Erase all objects in the workspace and list rest:

`rm()`

`ls()`

(**Basics)

Data types

Atomic (fundamental) Data Types?

complex:

raw:

numeric:

character:

logical:

Hierarchy?

Character > Double > Integer > Logical

(**Basics)

Loading Data

`read.table` txt data

`(file="name.txt")` frame

`read.csv2` csv data

`(file="name.csv")` frame

`read.spss` spss data

`(file="name.spss")` frame

`data` builtin

`(name, package="pkg")` dataset

`edit()` opens a spreadsheet

`read.file` library psy

`(file="name.spss")` ch import

functions.

(**Basics)

Parsing

A **parser** converts the textual representation of R code into an internal form which may then be passed to the R evaluator.

`(p1<-parse(text="3*(2+4)"))`

expression(3(2+4))*

`eval(p1)`

18

Abstract Syntax Trees (AST) that are parsed according to the language grammar

(**Advanced)

Parsing Image

`> lobstr::ast(3*(2+4))`



(**Advanced)

Vectors

Vectors two types:

Atomic: all elements same type

List: a generalized vector, elements can have different types

Ex: `vec <- c(1, 5, 4, 9, 0)`

Name the elements:

`(vec <- c(Frodo=1, Bilbo=5, Gandalf=9, 0))`

`length(x)`

3

Access the 1st element; remove the 3rd element

`vec[1]; vec[-3]`

Access the last element using the length

`vec[length(vec)]`

Access more elements

`vec[c(1,3)]`

Remove more elements

`vec[-c(1,3)]`

Access only elements that are greater than 3

`vec[vec>3]`

Replace one or more elements

`vec[c(1,3)] <- -c(1,3)`

Access only elements that are not NA

`vec[!is.na(vec)]`

Repeating

`rep(2,5) = c(2,2,2,2,2)`

Sequencing

`seq(2,5,1) 2,3,4,5`

Matrices are long Vectors

`matrix(1:4,2) + c(1,2)`

`which()`, `any()`, `all()`

R performs operations on vectors element by element

(**Basics)